THE INTEGRATION OF THE BACK PROPOGATION ALGORITHM INTO AN AUTONOMOUS ROBOT CONTROL SYSTEM

Under the guidance of: Professor M.V Deshpande

Abstract – This technical paper serves to provide a basic study into the field of Neural Networks. We will discuss the various architectures used in neural networks and provide an in-depth discussion into the Back Propagation Algorithm. Lastly, we will apply the back propagation algorithm to an autonomous control system to serve as an automatic guidance system. Neural networks can thus be applied in the control of motion of robot and serve as its automatic guidance system.

Index Terms—Neural Networks, Perceptron, Back Propagation, Robot, Path traversal, Autonomous Control System.

I. INTRODUCTION

This document serves to provide a formal documentation of our study in the field of Neural Networks. Over the course of this paper, we will focus on the back propagation algorithm in particular and focus on the integration of the back propagation algorithm into an autonomous control system of a robot.

1) Existing System

Our basic model is segregated into the following modules and functionality:

a. Creation of an external device using embedded technology.
b. The external device will be equipped with a control unit in the form of a microcontroller.
c. Equipping the devices with sensors to respond to its external environment.
d. Creation of multiple devices with varying processing power and functionality as per requirements.
e. Inter-connection of multiple devices using the computer which serves as an interface.
f. Wireless connectivity to remove the constraints of wires.
g. Creation of an autonomous device with self navigation capabilities thus implementing the essence of Artificial Intelligence.
h. The external devices will be aware of the relative progress of each other thus allowing Collaborative Thinking.
i. Information acquired by the devices is sent to a central server using wireless networking. Data processing at the server occurs to generate an accurate cartographic depiction of the territory traversed that spans three dimensions.

2) Proposed Enhancement

The proposed advancement is to implement a neural network coupled with back propagation and integrate it into the control system of an autonomous device. Thus, with the help of neural networking the device can learn, facilitating response to a dynamically changing environment.

II. NEURAL NETWORKS: AN INTRODUCTION

1) An Introduction to Neural Networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements.

We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this supervised learning, to train a network.
Batch training of a network proceeds by making weight and bias changes based on an entire set (batch) of input vectors. Incremental training changes the weights and biases of a network as needed after presentation of each individual input vector. Incremental training is sometimes referred to as "on line" or "adaptive" training.

3) Neuron Model – A Simple Neuron

A neuron with a single scalar input and no bias appears on the left below.

The scalar input \( p \) is transmitted through a connection that multiplies its strength by the scalar weight \( w \), to form the product \( wp \), again a scalar. Here the weighted input \( wp \) is the only argument of the transfer function \( f \), which produces the scalar output \( a \). The neuron on the right has a scalar bias, \( b \). You may view the bias as simply being added to the product \( wp \) as shown by the summing junction or as shifting the function \( f \) to the left by an amount \( b \). The bias is much like a weight, except that it has a constant input of 1.

The transfer function net input \( n \), again a scalar, is the sum of the weighted input \( wp \) and the bias \( b \). This sum is the argument of the transfer function \( f \). Here \( f \) is a transfer function, typically a step function or a sigmoid function, which takes the argument \( n \) and produces the output \( a \). Examples of various transfer functions are given in the next section. Note that \( w \) and \( b \) are both adjustable scalar parameters of the neuron. The central idea of neural networks is that such parameters can be adjusted so that the network exhibits some desired or interesting behavior. Thus, we can train the network to do a particular job by adjusting the weight or bias parameters, or perhaps the network itself will adjust these parameters to achieve some desired end.

III. LEARNING PARADIGMS

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. Usually any given type of network architecture can be employed in any of those tasks.

Supervised learning:

In supervised learning, we are given a set of example pairs \((x, y), x \in X, y \in Y\) and the aim is to find a function \( f \) in the allowed class of functions that matches the examples. In other words, we wish to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain.

A commonly used cost is the mean-squared error which tries to minimize the average error between the network's output, \( f(x) \), and the target value \( y \) over all the example pairs. When one tries to minimize this cost using gradient descent for the class of neural networks called Multi-Layer Perceptrons, one obtains the well-known back propagation algorithm for training neural networks.

Tasks that fall within the paradigm of supervised learning are pattern recognition (also known as classification) and regression (also known as function approximation). The supervised learning paradigm is also applicable to sequential data (e.g., for speech and gesture recognition).

Unsupervised learning:

In unsupervised learning we are given some data \( x \), and the cost function to be minimized can be any function of the data \( x \) and the network's output, \( f \).

The cost function is dependent on the task (what we are trying to model) and our \textit{a priori} assumptions (the implicit properties of our model, its parameters and the observed variables).
As a trivial example, consider the model \( f(x) = a \), where \( a \) is a constant and the cost \( C = (E[x] - f(x))^2 \). Minimizing this cost will give us a value of that is equal to the mean of the data. The cost function can be much more complicated. Its form depends on the application: For example in compression it could be related to the mutual information between \( x \) and \( y \). In statistical modeling, it could be related to the posterior probability of the model given the data. (Note that in both of those examples those quantities would be maximized rather than minimized)

Tasks that fall within the paradigm of unsupervised learning are in general estimation problems; the applications include clustering, the estimation of statistical distributions, compression and filtering.

**Reinforcement learning:**

In reinforcement learning, data \( x \) is usually not given, but generated by an agent's interactions with the environment. At each point in time \( t \), the agent performs an action \( y \), and the environment generates an observation \( x \), and an instantaneous cost \( c \), according to some (usually unknown) dynamics. The aim is to discover a *policy* for selecting actions that minimises some measure of a long-term cost, i.e. the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated.

IV. ARCHITECTURES OF NEURAL NETWORKS

1) Feed-forward networks

Feed-forward Artificial Neural Networks allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward Artificial Neural Networks tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down

![Feed-forward Networks](image1)

2) Feedback networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations.

![A Complex Neural Network](image2)

V. TRANSFER FUNCTION

The behavior of an Artificial Neural Network depends on both the weights and the input-output function (transfer function) that is specified for the units. This function typically falls into one of three categories:

- **Linear units**: The output activity is proportional to the total weighted output.

- **Threshold units**: The output is set at one of two levels, depending on whether the total input is greater than or less than some threshold value.

- **Sigmoid units**: The output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered rough approximations.

To make a neural network that performs some specific task, we must choose how the units are connected to one another and we must set the weights on the connections appropriately. The connections determine whether it is possible for one unit to influence another. The weights specify the strength of the influence.
We can teach a three-layer network to perform a particular task by using the following procedure:

1. We present the network with training examples, which consist of a pattern of activities for the input units together with the desired pattern of activities for the output units.

2. We determine how closely the actual output of the network matches the desired output.

3. We change the weight of each connection so that the network produces a better approximation of the desired output.

4. BACK PROPAGATION NETWORKS

A back propagation network is a fully connected, layered, and feed-forward neural network (see Figure a). Network activation flows in one direction only: from the input layer to the output layer, passing through the hidden layer. Each unit in a layer is connected in the forward direction to every unit in the next layer. Weights between units encode the network's knowledge.

After each round of forward-backward passes, the system "learns" incrementally from the input-output pair and reduces the difference (error) between the network's predicted output and the actual output. After extensive training, the network will eventually establish the input-output relationships through the adjusted weights on the network.

VI. THE BACK PROPAGATION ALGORITHM

Given a set of input-output vector pairs, you can compute a set of weights for a neural network that maps inputs onto corresponding outputs.

Let A be the number of units in the input layer, as determined by the length of the training input vectors. Let C be the number of units in the output layer. Now choose B, the number of units in the hidden layer. As shown in Figure a, the input and hidden layers each have an extra unit used for thresholding; therefore, the units in these layers will sometimes be indexed by the ranges (0,...,A) and (0,..., B). We denote the activation levels of the units in the input layer by \(x_i\), in the hidden layer by \(h_i\), and in the output layer by \(o_j\). Weights connecting the input layer to the hidden layer are denoted by \(w_{1ij}\), where the subscript \(i\) indexes the input units and \(j\) indexes the hidden units. Likewise, weights connecting the hidden layer to the output layer are denoted by \(w_{2ij}\), with \(i\) indexing hidden units and \(j\) indexing output units.

The back propagation algorithm has the following steps:

1. Initialize the network weights. Initially, all connection weights are set randomly to numbers between -0.1 and 0.1:
   \[
   w_{1ij} = \text{random( -0.1, 0.1 )} \quad \text{for all } i = 0, ..., A, j = 1, ..., B \\
   w_{2ij} = \text{random( -0.1, 0.1 )} \quad \text{for all } i = 0, ..., B, j = 1, ..., C
   \]

2. Initialize the activations of the threshold units. For each layer, its threshold unit is set to 1 and should never change:
   \[
   x_0 = 1.0 \\
   h_0 = 1.0
   \]

3. Choose an input-output pair. Suppose the input vector is \(x_i\) and the target output vector is \(y_i\). Assign activation levels to the input units.

4. Propagate the activations from the units in the input layer to the units in the hidden layer using the activation function:
\[ h_j = \frac{1}{1 + e^{-\sum_{i=0}^{A} w_{1ij} h_i}} \]

for all \( j = 1, \ldots, B \)

5. Note that \( i \) ranges from 0 to \( A \). \( w_{1ij} \) is the thresholding weight for hidden unit \( j \). \( x_0 \) is always 1.0.

6. Propagate the activations from the units in the hidden layer to the units in the output layer:

\[ o_j = \frac{1}{1 + e^{-\sum_{i=0}^{B} w_{2ij} h_i}} \]

for all \( j = 1, \ldots, C \)

7. Again, the thresholding \( w_{2oj} \) for output units \( j \) plays a role in the weighted summation. \( h_0 \) is always 1.0.

8. Compute the errors of the units in the output layer, denoted \( \delta_2 \). Errors are based on the network's actual output \( o_j \) and the target output \( y_j \):

\[ \delta_2 = o_j (1 - o_j) (y_j - o_j) \text{ for all } j = 1, \ldots, C \]

9. Compute the errors of the units in the hidden layer, denoted \( \delta_1 \):

\[ \delta_1_j = h_j (1 - h_j) \sum_{i=1}^{C} \delta_2_i w_{2ji} \]

for all \( j = 1, \ldots, B \)

10. Adjust the weights between the hidden layer and output layer. The learning rate is denoted \( \eta \); its function is the same as in perceptron learning. A reasonable value of \( \eta \) is 0.35:

\[ \Delta w_{2ij} = \eta \delta_2_j h_i \text{ for all } i = 0, \ldots, B, j = 1, \ldots, C \]

11. Adjust the weights between the input layer and the hidden layer:

\[ \Delta w_{1ij} = \eta \delta_1_j x_i \text{ for all } i = 0, \ldots, A, j = 1, \ldots, B \]

12. Go to Step 4 and repeat. When all the input-output pairs have been presented to the network, one epoch has been completed. Repeat Steps 4 to 10 for as many epochs as desired.

The activation function has a sigmoid shape. Since infinite weights would be required for the actual outputs of the network to reach 0.0 and 1.0, binary target outputs (the \( y_j \)'s of Steps 4 and 7 above) are usually given as 0.1 and 0.9 instead. The sigmoid is required by backpropagation because the derivation of the weight update rule requires that the activation function be continuous and differentiable.

VII. INTEGRATION OF BACK PROPAGATION ALGORITHM WITH AN AUTONOMOUS CONTROL SYSTEM

To define input-output vector pairs for use in the backpropagation network, from the robot input-output (sensor-motor), we must identify what the robot is going to learn.

![Backpropagation network coupled with sensors and motors](image)
We define four basic behavior rules:

a) **Moving forward**: If Sensor 1 is off, and Sensor 2 is over a white floor, and Sensor 3 is off, then Motor A and Motor C go forward (Robot goes forward)

b) **Moving right**: If Sensor 1 is on, then Motor A goes forward, and Motor C goes backward (Robot turns right)

c) **Moving left**: If Sensor 3 is on, then Motor A goes backward, and Motor C goes forward (Robot turns left)

d) **Moving backward**: If Sensor 2 is over a black floor, then Motor A and Motor C go backward (Robot goes backward)

We translate these rules to training examples for the backpropagation network as shown, where S1 = Sensor 1, M-A = Motor A, and so on.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Sensor 1</th>
<th>Sensor 2</th>
<th>Sensor 3</th>
<th>Motor A</th>
<th>Motor C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Off</td>
<td>White</td>
<td>Off</td>
<td>Forward</td>
<td>Forward</td>
</tr>
<tr>
<td>2</td>
<td>On</td>
<td>White</td>
<td>Off</td>
<td>Forward</td>
<td>Backward</td>
</tr>
<tr>
<td>3</td>
<td>Off</td>
<td>White</td>
<td>On</td>
<td>Backward</td>
<td>Forward</td>
</tr>
<tr>
<td>4</td>
<td>Off</td>
<td>Black</td>
<td>Off</td>
<td>Backward</td>
<td>Backward</td>
</tr>
</tbody>
</table>

The input-output vector pairs are the examples we use to train the backpropagation network. So, based on its sensor states, our robot will learn to move forward, right, left, and backward.

**VIII. Applications of Neural Networks**

Given this description of neural networks and how they work, what real world applications are they suited for? Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including:

**Aerospace**: High performance aircraft autopilot, flight path simulation, aircraft control systems, autopilot enhancements, aircraft component simulation, aircraft component fault detection

**Automotive**: Automobile automatic guidance system, warranty activity analysis

**Banking**: Check and other document reading, credit application evaluation

**Credit Card Activity Checking**: Neural networks are used to spot unusual credit card activity that might possibly be associated with loss of a credit card

**Defense**: Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification

**Electronics**: Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling

**Entertainment**: Animation, special effects, market forecasting

**Financial**: Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit-line use analysis, portfolio trading program, corporate financial analysis, currency price prediction

**Industrial**: Neural networks are being trained to predict the output gasses of furnaces and other industrial processes.
They then replace complex and costly equipment used for this purpose in the past.

**Insurance**: Policy application evaluation, product optimization

**Manufacturing**: Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer-chip quality analysis, analysis of grinding operations, chemical product design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process system

**Medical**: Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency-room test advisement

**Robotics**: Trajectory control, forklift robot, manipulator controllers, vision systems

**Speech**: Speech recognition, speech compression, vowel classification, text-to-speech synthesis

**Securities**: Market analysis, automatic bond rating, stock trading advisory systems

**Telecommunications**: Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems

**Transportation**: Truck brake diagnosis systems, vehicle scheduling, routing systems

**IX. CONCLUSION**

The following technical paper gives insight into the basics of Neural Networking. It then goes on to focus on the Back Propagation Algorithm and its integration with an autonomous control system. It serves as a supplement to an existing paper titled, ‘Autonomous Robotics’ which is based upon an existing final year project that is carried out as part of our curriculum for the Bachelor of Engineering.

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that some day 'conscious' networks might be produced. There are a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' neural networks are a realistic possibility.

Using Neural Networking, an autonomous device can learn while traversing unmapped territory. It can thus constantly update its knowledge base using the back propagation algorithm, thus facilitating response to a dynamically changing environment.

**X. ACKNOWLEDGMENT**

We would like to take this opportunity to thank the Computer Department at DJ Sanghvi College of Engineering for providing us with the necessary resources needed for the successful implementation of the project.

We would like to thank Ms. Kiran Bhoumik who is our co-guide for our final year B.E Project. I would also like to express my gratitude to Professor M.V Deshpande, who in his capacity as project guide and head of department, provided us with the guidance and helping hand needed in the successful deployment of the project.

**XI. APPENDIX A – NEURAL NETWORKS IN MEDICINE**

Artificial Neural Networks (ANN) is currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modeling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.).

Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

**Modeling and Diagnosing the Cardiovascular System:**

Neural Networks are used experimentally to model the human cardiovascular system. Diagnosis can be achieved by building a model of the cardiovascular system of an individual and comparing it with the real time physiological measurements taken from the patient. If this routine is carried out regularly, potential harmful medical conditions can be
detected at an early stage and thus make the process of combating the disease much easier.

A model of an individual's cardiovascular system must mimic the relationship among physiological variables (i.e., heart rate, systolic and diastolic blood pressures, and breathing rate) at different physical activity levels. If a model is adapted to an individual, then it becomes a model of the physical condition of that individual. The simulator will have to be able to adapt to the features of any individual without the supervision of an expert. This calls for a neural network.

Another reason that justifies the use of ANN technology, is the ability of ANNs to provide sensor fusion which is the combining of values from several different sensors. Sensor fusion enables the ANNs to learn complex relationships among the individual sensor values, which would otherwise be lost if the values were individually analysed. In medical modelling and diagnosis, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, ANNs are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors.

Electronic noses:

ANNs are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link. The electronic nose would identify odors in the remote surgical environment. These identified odors would then be electronically transmitted to another site where an door generation system would recreate them. Because the sense of smell can be an important sense to the surgeon, telesmell would enhance surgery.

Instant Physician:

An application developed in the mid-1980s called the "instant physician" trained an auto associative memory neural network to store a large number of medical records, each of which includes information on symptoms, diagnosis, and treatment for a particular case. After training, the net can be presented with input consisting of a set of symptoms; it will then find the full stored pattern that represents the "best" diagnosis and treatment.

XII. APPENDIX B – NEURAL NETWORKS IN BUSINESS

Business is a diverted field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis. There is some potential for using neural networks for database mining, that is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling.

Marketing:

There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feed forward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution. The system is used to monitor and recommend booking advice for each departure. Such information has a direct impact on the profitability of an airline and can provide a technological advantage for users of the system. [Hutchison & Stephens, 1987]

While it is significant that neural networks have been applied to this problem, it is also important to see that this intelligent technology can be integrated with expert systems and other approaches to make a functional system. Neural networks were used to discover the influence of undefined interactions by the various variables. While these interactions were not defined, they were used by the neural system to develop useful conclusions. It is also noteworthy to see that neural networks can influence the bottom line.

Credit Evaluation:

The HNC Company, founded by Robert Hecht-Nielsen, has developed several neural network applications. One of them is the Credit Scoring system which increases the profitability of the existing model up to 27%. The HNC neural systems were also applied to mortgage screening. A neural network automated mortgage insurance underwriting system was developed by the Nestor Company. This system was trained with 5048 applications of which 2597 were certified. The data related to property and borrower qualifications. In a conservative mode the system agreed on the underwriters on 97% of the cases. In the liberal model the system agreed 84% of the cases. This is system run on an Apollo DN3000 and used 250K memory while processing a case file in approximately 1 sec.

XIII. REFERENCES

[2] Neural Networks by Eric Davalo and Patrick Naim